|  |  |
| --- | --- |
| File Processing Programming Project #1 | |
| 20121592 박재혁 | |
| 1. Definition and requirement for implementation | |
| 1. Basic Class | |
| 1. .Scheme class | |
|  | virtual function을 통해, 자식 Class가 될 Basic Class에 대해 간략한 interface를 제공한다. |
| .2> Member class | |
| Data 영역 | Method 영역 |
| std::string ID;  std::string Password;  std::string Name;  std::string PhoneNumber;  std::string Address;  char Mileage[MILEAGE\_LEN +1]; | Member();  Member(const Scheme& s);  //생성자, 복사 생성자 포함  Member& operator =(const Member &s)  bool operator ==(const Member &s)  bool operator !=(const Member &s)  //연산자 오버로딩  Member& operator =(const Scheme &s);  bool operator ==(const Scheme &s);  bool operator !=(const Scheme &s);  //부모 클래스를 좌변으로 한 오버로딩  void update\_ID(const std::string ID)  void update\_Password(const std::string Password)  void update\_Name(const std::string Name)  void update\_PhoneNumber(const std::string PhoneNumber)  void update\_Address(const std::string Address)  void update\_mileage(const char\* new\_mileage)  //Data 업데이트 함수  friend istream &operator >> (istream& is, Member &s);  friend ostream &operator << (ostream& os, Member &s);  //stream 연산자 오버로딩  bool Pack(IOBuffer& Buffer) const;  bool Unpack(IOBuffer &); |
| 예제 테이블 |
|  |
| .3> Lecture class | |
| Data 영역 | Method 영역 |
| char LectureID[LECID\_LEN+1];  std::string Subject;  char Level;  int Price;  char Extension;  int Due\_date;  std::string Name\_of\_teacher;  std::string Textbook; | Lecture();  Lecture(const Scheme& s);  //생성자, 복사 생성자 포함  Lecture& operator =(const Scheme &s);  bool operator ==(const Scheme &s);  bool operator !=(const Scheme &s);  //연산자 오버로딩  Lecture& operator =(const Lecture &s)  bool operator ==(const Lecture &s)  bool operator !=(const Lecture &s)  //부모 클래스를 좌변으로 한 오버로딩  void update\_lectureid(std::string new\_lectureid)  void update\_subject(std::string new\_subject)  void update\_level(std::string new\_level)  void update\_price(std::string new\_price)  void update\_extension(std::string new\_extension)  inline void update\_duedate(std::string new\_duedate)  void update\_nameofteacher(std::string new\_nameofteacher)  void update\_textbook(std::string new\_textbook)  //Data 업데이트 함수  friend istream &operator >> (istream& is, Lecture &s);  friend ostream &operator << (ostream& os, Lecture &s);  //stream 연산자 오버로딩  bool Pack(IOBuffer& Buffer) const;  bool Unpack(IOBuffer &); |
| 예제 테이블 |
|  |
| .4> Purchase class | |
| Data 영역 | Method 영역 |
| std::string PurchaseID;  char LectureID[LECID\_LEN + 1];  std::string MemberID;  char Mileage[MILEAGE\_LEN + 1]; | Purchase();  Purchase(const Scheme& s);  //생성자, 복사 생성자 포함  Purchase& operator =(const Purchase &s)  bool operator ==(const Purchase &s)  bool operator !=(const Purchase &s)  //연산자 오버로딩  Purchase& operator =(const Scheme &s);  bool operator ==(const Scheme &s);  bool operator !=(const Scheme &s);  //부모 클래스를 좌변으로 한 오버로딩  void update\_purchaseid(std::string new\_Purchaseid)  inline void update\_lectureid(std::string new\_lectureid)  inline void update\_memberid(const std::string new\_memberid)  inline void update\_mileage(const char\* new\_mileage)  //Data 업데이트 함수  friend istream &operator >> (istream& is, Purchase &s);  friend ostream &operator << (ostream& os, Purchase &s);  //stream 연산자 오버로딩  bool Pack(IOBuffer& Buffer) const;  bool Unpack(IOBuffer &); |
| 예제 테이블 |
|  |
| 1. Definition and requirement for implementation | |
| 1. 템플릿을 두어 추상 메소드를 만들고, 각 Basic Class에 따라서 적용 후 function table에 놓고 필요한 작업을 수행한다. 2. 노란색으로 칠한 부분을 위해, <<와 >>의 오버로드 연산자를 구현하였다. | |
| 1. Using IOBuffer | |
| 1. Basic Class | |
| (1) 템플릿을 두어 추상 메소드를 만들고, 각 Basic Class에 따라서 적용 후 function table에 놓고 필요한 작업을 수행한다.  (2) Write\_Unique라는 함수를 구현함으로써, 중복된 레코드는 .Dat File에 쓰지 않는다.    (3) Write\_Unique의 내부 구현은 현재 가지고 있는 record가 File 내에 존재하는 지 찾은 뒤, 없으면 쓰는 방식으로 구현되어 있다..     1. Find는 두 가지 방식으로 parameter에 따른 오버로딩이 되어있다.위는 인자로 넘긴 오브젝트와 일치하는 레코드를 파일 내에서 찾아서 오프셋을 반환한다. 아래는 index를 통해 index에 해당하는 값을 찾고, 파일에 있는 record를 메모리 상의 오브젝트로 변환하여 반환한다.      1. 기존의 인풋 스트림, 아웃풋 스트림의 오프셋을 저장하고 있다가, 전체 레코드를 Read를 통해 선형 탐색을 한다. 시간 복잡도는 약 O(n)이다. 저장된 인풋 스트림과 아웃풋 스트림은 탐색이 종료됨과 동시에 다시 File에 돌려준다. | |
| 1. Dat Data 내의 저장 방식과 Read, Write, Delete | |
| 1. Dat Data 내의 저장 방식 | |
| 1. 기존 Record가 삭제되었을 때, Record의 형식이 다음과 같이 바뀐다. 2. Record File 내의 헤더는 다음과 같이 변경되었다.      1. 본문에서 가변 레코드는 다음과 같은 방식으로 필요한 만큼의 페이지를 얻어 그 안에 할당하는 식으로 구성되어있다. 2. 한 페이지의 단위는 128byte로 구성했으며, 모든 offset은 128의 배수로 움직인다. | |
| 1. Read, Write, Delete | |
| 1> Read     1. 해당 offset에서 페이지 개수에 해당하는 앞쪽 헤더만 추출하여, record가 총 몇 page를 차지하고 있는지 파악한다. 2. 페이지수만큼의 buffer를 준비하여 그대로 복사한다. 3. 이 때 앞 offset이 0이면, 삭제된 레코드이므로 offset을 한 페이지만큼 넘어가고 (1)의 작업을 반복한다.   2> Write     1. 파일의 끝 부분이면, 현재 가지고 있던 Pack된 버퍼의 크기를 수용할 수 있는 최소 단위의 페이지를 할당하여, Record에 쓴다.      1. 삭제된 레코드 위에서는, 삭제된 레코드가 연속되어 몇 개가 존재하고 있는지 확인한 뒤 버퍼 크기가 연속된 페이지의 총합보다 작으면, Record에 쓴다.   3> Delete     1. 현재 offset에서 맨 앞에 있는 Page 개수를 확인한 뒤, 그 개수만큼 일련의 페이지당 삭제레코드로 교체해준다. | |
| 1. Class Diagram | |
|  | |
| 1. Deleting and Updating Records | |
| 연습문제 답안 | |
| 21, 23번)  21번과 23번을 해결하기 위해, 4번의 저장방식을 택했다.  VariableLengthBuffer의 Read, Write, Delete의 수정을 통해 구현했다.  22, 24번)  삭제된 레코드를 재사용될 수 있도록, 삭제된 레코드에 이전 삭제 레코드와 다음 삭제 레코드의 오프셋을 저장할 수 있는 double list 형태의 자료구조를 구현하였다. 5번 class diagram의 recordform.hpp에 다음의 구현을 돕는 메소드들이 있다. 25번)  새로 추가되는 레코드가 삭제되는 레코드보다 클 경우를 대비하여, 가변 레코드를 페이지 방식으로 저장되있고 연속되어있으면서 삭제된 페이지 개수를 비교하여 새로운 레코드를 쓸 수 없을지 판별할 수 있다. 해당 레코드가 더 작은 경우 file의 맨 뒤에 append하거나 다른 빈 공간을 찾을 수 있다  26번)  b., 페이지 단위로 저장하여 원천 차단했다. | |
| 검색, 삽입, 삭제, 수정 방법 | |
| 검색 : 2. 에서 설명한 Find 함수를 통해 찾는다.  삽입 : 3. 선형 탐색으로 삽입하고자 하는 레코드가 차지하는 페이지보다 연속된 삭제 레코드 크기가 더 큰 페이지들의 집합을 찾아서 삽입한다. (compaction)  삭제 : Find 함수를 통해 삭제하고자 하는 레코드의 Offset을 얻은 뒤 IO Buffer의 Delete 기능으로 삭제한다.  수정 : Find 함수를 통해 삭제하고자 하는 수정하고자 하는 Offset을 얻은 뒤, 원본 레코드를 삭제하고 위에서 제시한 삽입 방법으로 새로 수정된 레코드를 넣는다.  Find가 선형 탐색이므로 검색하는 데 O(n), 삽입할 곳 찾는데 O(n)으로, 선형 시간 안에 검색, 삽입, 삭제, 수정을 끝낼 수 있다. | |